개방형 GIS 컴포넌트의 성능 개선을 위한 버퍼 관리 방법의 설계 및 구현

조 대 수[†] · 민 경 욱[††]

요 약

개방형 GIS에서 GIS 클라이언트는 동일한 인터페이스를 통해 서로 다른 GIS 서버의 공간 데이터에 접근할 수 있는 장점을 갖는다. 따라서, 개방형 GIS 컴포넌트 소프트웨어는 서로 다른 GIS 서버간의 상호운용성(interoperability)을 보장한다. 그러나, GIS 클라이언트가 공간 데이터에 접근하기 위해서 표준 인터페이스를 사용한다면, 사용자의 응답시간이 느려지는 단점을 갖는다. 왜냐하면, 상호운용성을 위해서는 각 GIS 서버로부터 접근된 공간 데이터를 표준 인터페이스를 통해 접근할 수 있도록 OLE/DB의 로셋과 같은 공통된 데이터 모델로 변환해야 하기 때문이다. 이 논문에서는 GIS 클라이언트에서 데이터 버퍼를 통해 사용자 응답 시간의 지연 문제를 해결한다. 이 논문에서는 공간 분할에 의한 버퍼 관리 방법을 설계하고, 개방형 GIS 컴포넌트 소프트웨어인 MapBase 컴포넌트를 통해 구현하였다. 또한 성능 평가 실험을 통해 이 논문에서 제안한 버퍼 관리 방법이 GIS 클라이언트에서 데이터 접근 성능을 개선함을 보였다.

Design and Implementation of Buffer Management Method for Enhancing Performance of Open GIS Components

Dae-Soo Cho[†] · Kyoung-Wook Min[††]

ABSTRACT

In open GIS environment, a GIS client can access spatial data in different types of GIS servers with the same interfaces. This means that open GIS components software ensures the interoperability throughout the heterogeneous GIS servers. The user response time, however, tends to be increased, if the client makes use of the standard interfaces for data accesses that can ensure interoperability. This is because the format of spatial data accessed from a specific GIS server must be transformed into common format, such as Rowset in OLE/DB, which is compatible with the standard interfaces. In this paper, we develop efficient techniques for data buffering in GIS client to reduce the response time. We design the buffer management method, which is based on the space partitioning, and integrate buffer management components into MapBase, an open GIS component software. And we also, show that buffer management proposed in this paper yields significant performance improvement in GIS client.

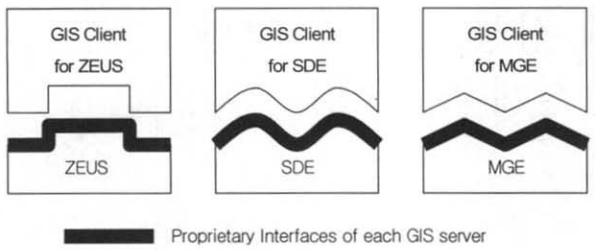
키워드 : 개방형 GIS(Open GIS), 메모리 관리(Memory Management), 버퍼 관리(Buffer Management), 공간데이터베이스(Spatial Database)

1. 서 론

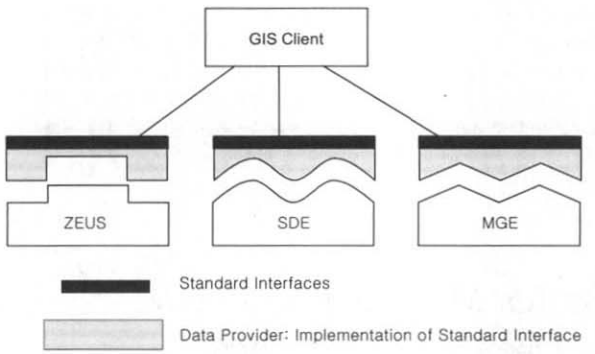
지리정보시스템(GIS, Geographic Information System)은 전통적인 클라이언트/서버 구조에서 표준 인터페이스를 갖는 개방형 구조로 발전하고 있다. (그림 1)(a)는 기존의 클라이언트/서버 구조의 GIS를 보이고 있다. 각각의 GIS 서버는 자신만의 고유한 인터페이스(Proprietary Interfaces)를 갖고 있으므로, 각 GIS 서버별로 별도의 GIS 클라이언트를 필요로 한다. 따라서, 클라이언트/서버 GIS에서 하나의 클라이언트는 서로 다른 GIS 서버의 공간 데이터에 접근할 수 없기 때문에, 각각의 GIS 수요기관에서 서로 다른 GIS 서버로 구축된 방대한 양의 공간 데이터를 서로 공유하기가 어려운 문제점을 갖는다.

방대한 양의 공간 데이터를 공유하기 위해서, 서로 다른 GIS 서버간의 상호운용성(Interoperability)을 제공하는 개방형 GIS 구조는 (그림 1)(b)와 같다. 개방형 GIS에서 클라이언트는 동일한 방법으로 서로 다른 GIS 서버의 공간 데이터에 접근할 수 있다. 동일한 방법이란 GIS 클라이언트가 GIS 서버의 종류에 상관없이 표준 인터페이스(Standard Interface)를 이용하여 데이터에 접근하는 것을 의미한다. 따라서, 개방형 GIS에서 각각의 GIS 서버는 자신만의 고유한 인터페이스를 감싸며, 표준 인터페이스를 노출해 주는 데이터제공자(Data Provider)를 제공해야 한다.

† 정 회 원 : 한국전자통신연구원 LBS 연구팀 선임연구원
†† 정 회 원 : 한국전자통신연구원 LBS 연구팀 연구원
논문접수 : 2002년 7월 4일, 심사완료 : 2003년 9월 24일



(a) 클라이언트/서버 GIS 구조



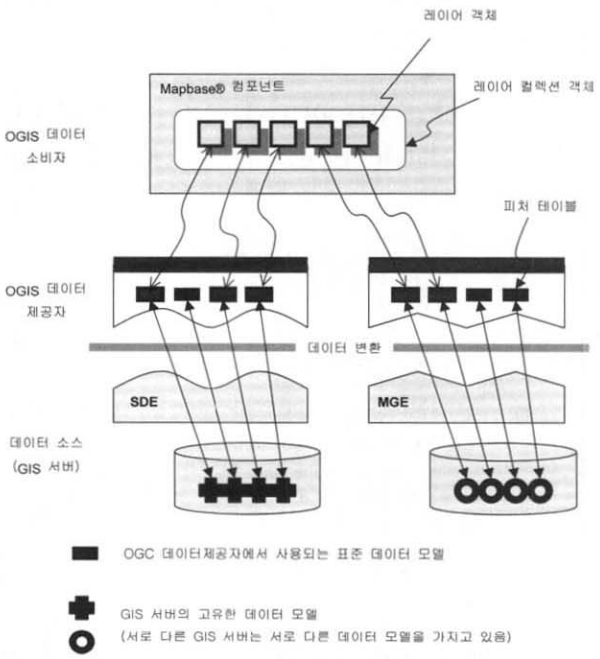
(b) 개방형 GIS 구조

(그림 1) 지리정보시스템(GIS)의 구조

GIS 분야에서 개방형 구조를 위한 인터페이스의 표준화는 OGC(Open GIS Consortium)를 통해 활발히 진행되고 있으며, 국제 표준화 기구인 ISO의 19125를 통해 국제 표준으로 제정되기 위한 절차가 진행되고 있다. 따라서, 이 논문에서의 개방형 GIS 컴포넌트는 OGIS 데이터소비자(클라이언트를 의미함)가 OGIS 데이터제공자(OGC의 표준 인터페이스에 따라 구현된 데이터제공자를 의미함)를 통해 해당 GIS 서버에 접근하는 것을 의미한다.

OGC에서 제안하는 표준 인터페이스인 심플 피처 구현 명세는 OGIS 데이터제공자를 구현하는 방법에 따라서 크게 세가지가 있으며, 이 논문에서 OGIS 데이터제공자는 OLE/COM를 위한 심플 피처 구현 명세[12]에 따라서 구현된 것을 의미한다. OGIS 데이터제공자는 마이크로소프트(Microsoft)의 OLE DB 인터페이스를 기반으로 하고 있으며, 기하 데이터(Geometry) 및 공간참조체계(Spatial Reference System) 같이 GIS에서 필요한 정보등에 접근하거나, 공간 연산(Spatial Operator)을 수행하기 위해서 추가의 인터페이스를 정의한 것이다.

(그림 2)는 OGIS 데이터소비자가 표준 인터페이스를 제공하는 OGIS 데이터제공자를 통해서 서로 다른 GIS 서버의 공간 데이터에 접근하는 것을 보이고 있다. OGIS 데이터제공자는 OGC의 OLE/COM을 위한 심플 피처 구현 명세[12]에 따라 구현되었기 때문에, 연결된 GIS 서버의 종류에 관계없이 피처 테이블(Feature Table) 형태로 공간 데이터에 접근할 수 있도록 각 GIS 서버의 공간 데이터를 변환한다. 따라서, OGIS 데이터소비자가 특정 GIS 서버의 공간 데이터에 접근하기 위해서는 내부적으로 두 번의 연결과정과 두 번의 데이터 접근과정이 필요하므로, 데이터 접근 속도가 느려지게 된다. 특히, 공간 데이터를 구성하는 속성 데이터(Attribute)와 기하 데이터(Geometry) 중에서 상대적으로 양이 많은 기하 데이터의 경우에는 WKB(Well Known Binary) 형태로 변환되어 피처 테이블의 하나의 열(Column)에 저장되기 때문에, 속성 데이터에 비해서 더욱 복잡한 과정을 거쳐서 접근해야 하는 문제가 있다. 따라서 개방형 GIS 구조는 OGIS 데이터소비자가 각 GIS 서버에 있는 공간/비공간 데이터에 접근하는 속도가 저하되는 구조적인 문제를 갖는다. 대신에 이러한 데이터 변환 비용을 통해 서로 다른 GIS 서버간의 상호 운용성이 달성되는 것이다.



(그림 2) OGIS 데이터제공자를 통한 서로 다른 GIS 서버로부터의 공간데이터 접근

이 논문은 개방형 GIS 구조에서 공간 데이터의 버퍼 관리(Buffer Management)를 다룬다. 버퍼 관리는 OGIS 데이터소비자의 느린 데이터 접근 속도 문제를 간접적으로 해결할 수 있는 방법이다. 개방형 GIS 구조에서 공간 데이터의 버퍼 관리가 어려운 이유는 첫째, 공간 데이터의 양이 매우 크기 때문이며, 둘째, GIS 서버에서 관리되던 공간 데이터에 대한 메타 정보 및 공간 인덱스 등이 OGIS 데이터제공자를 거치면서 모두 사라지기 때문이다.

이 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대해서 살펴보고, 3장에서는 개방형 GIS 컴포넌트에서의 버퍼 관리의 문제점과 이 논문에서 제안하는 해결 방법을 설명한다. 4장에서는 개방형 GIS 컴포넌트인 MapBase 컴포

넌트에서 버퍼 관리 시스템의 설계를 설명한다. 5장에서는 버퍼 관리 시스템 구현 및 성능 평가에 대해 설명하고, 6장에서 결론을 맺는다.

2. 관련 연구

데이터 접근 비용의 개선을 위한 연구로는 (A)메모리 버퍼(또는 캐쉬) 관리 방법에 대한 연구와 (B)데이터 프리페치에 관한 연구가 있다. 메모리 버퍼에 관한 연구에는 메모리 버퍼의 적중률을 높이기 위한 새로운 (A-1)버퍼 교체 전략에 대한 연구와 버퍼된 데이타를 효과적으로 관리하기 위해 (A-2)버퍼 구조에 대한 연구가 있다. 메모리 버퍼에 대한 연구는 운영 체제나 DBMS를 대상으로 많이 이루어지고 있으며, 특정 응용 분야(예를 들어 CAD와 같이 데이타 집중적인 응용 분야)에서도 디스크 I/O를 줄이기 위해 많이 이루어지고 있다.

대부분의 응용에서 가장 널리 사용되는 버퍼 교체 전략은 LRU(Least Recently Used), LFU(Least Frequently Used) 등[3]이 있으며, 특히 LRU 교체 전략이 구현의 단순함과 우수한 성능으로 인해 상용 시스템에서 가장 많이 구현되고 있다. 이러한 LRU 교체 전략에 대한을 개선하기 위한 연구에서는 기존의 LRU 교체 전략에 대해서 다음의 두 가지 문제점을 제시하고 있다[4]. 첫째는 버퍼 데이타를 교체할 때 이용되는 정보가 너무 적다는 점이다. LRU 교체 전략에서는 가장 최근에 접근된 시간만을 이용해서 버퍼된 데이타를 교체하기 때문에 특정 응용에 따라 추가로 제공될 수 있는 정보를 활용하지 못한다는 약점이 있다. 둘째는 접근 빈도(frequency)가 다른 데이타를 동일하게 다룬다는 점이다. 자주 사용되는 데이타도 자주 사용되지는 않지만 최근에 사용된 데이타에 비해 교체될 우선순위가 높다는 문제점을 가지고 있다. 따라서 특정 응용 분야에서 추가로 이용할 수 있는 정보로써 사용자의 데이타 접근 유형[16], 접근 빈도[4, 6, 11], 객체의 시맨틱[13] 등을 이용하는 버퍼 교체 전략 등이 제시되었다. 특히, 시간만을 고려하는 LRU의 특성에 최근의 접근 빈도수를 함께 고려한 LRU/K[3], 2Q[11] 교체 전략이 특정 응용에서 우수한 성능을 보였다.

메모리 버퍼 구조에 대한 연구에서는 버퍼의 적중률을 높이기 위한 교체 전략 알고리즘을 제시하는 것이 아니라, 메모리 버퍼의 공간 활용도(utilization)를 높이거나 데이타 페이지의 클러스터링 정보를 이용하여 디스크 I/O를 줄이는 방법을 제안하고 있다. 대표적인 연구로는 기존의 객체-지향 버퍼와 페이지-지향 버퍼의 구조를 혼합한 이중 버퍼(dual-buffering) 알고리즘[2]이 있다. 이중 버퍼 알고리즘과 같이 서로 다른 두 개의 메모리 버퍼 구조를 갖지 않더라도 동일한 종류의 버퍼 데이터를 그 특성에 따라 분리해서 관리하는 방법[6, 11]도 기존의 연구에서 많이 활용되고 있다.

데이터 접근 비용을 개선하기 위한 또 다른 연구 분야인 데이터 프리페치에 대한 연구에서는 주로 프리페치의 대상을 결정하기 위한 알고리즘을 제시하고 있다. 즉, 프리페치될 후보 데이타를 결정하는 데에는 접근 빈도[6], 현재 접근되는 객체와의 관련성[6, 13], 과거 사용자의 데이타 접근 유형을 분석 결과[10] 등이 이용되고 있다.

이 논문은 개방형 GIS 컴포넌트에서 메모리 및 디스크 버퍼의 구조를 제안한고, 구현하는 것을 목적으로 한다. 따라서, 새로운 버퍼 교체 전략이나, 프리페치 알고리즘은 이 논문의 대상이 아니다. 이 논문에서는 구현한 버퍼 관리 시스템에서 버퍼 교체 전략은 기존의 LRU를 이용하였다.

3. 문제점 및 해결방법

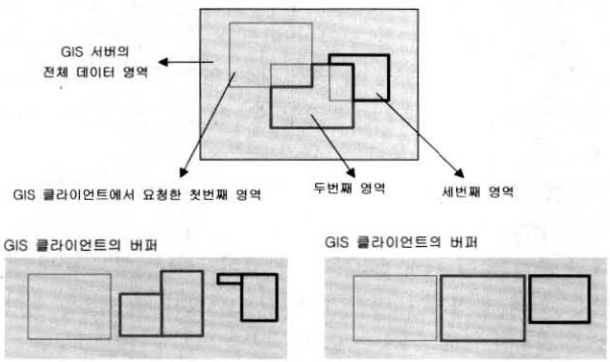
이 장에서는 개방형 GIS 컴포넌트에서 버퍼 관리 시스템을 구현하기 위해 해결해야 할 문제점과 이를 해결하기 위한 이 논문의 접근 방법을 설명한다. 먼저 이 논문에서 제안하는 버퍼 관리 방법의 범위는 다음과 같다. 첫째, 버퍼 관리의 대상은 GIS 클라이언트에서 화면 출력을 위해 영역 질의를 통해 접근한 공간 데이터이다. 즉, 영역 질의 이외의 요청에 의해 접근한 공간 데이터는 버퍼 관리의 대상이 아니다. 왜냐하면, 화면 출력을 위한 영역 질의는 GIS에서 가장 빈번히, 중복적으로 발생될 뿐 아니라, 전송되는 데이터의 양도 상대적으로 매우 많기 때문이다.

둘째, 이 논문에서는 버퍼에 관리되는 공간 데이터의 일관성 문제를 다루지 않는다. 왜냐하면, 버퍼 관리의 대상이 화면 출력을 위해 요구되는 공간 데이터이며, 대개 화면 출력을 위해 요청되는 공간 데이터는 반드시 최신성을 유지하지 않아도 되는 특징을 갖기 때문이다. 즉, 메모리 및 디스크 버퍼의 내용은 주변 지역에 대한 참조로 사용될 뿐이며, 공간 질의를 통한 분석 작업은 최신 공간 데이터를 보유하고 있는 GIS 서버에서 수행된다.

3.1 버퍼 관리 단위

버퍼 관리는 객체 단위 또는 페이지 단위로 이루어질 수 있으나[2], 이 논문에서 다루려는 버퍼 관리 방법에서는 사용하기 어렵다. 객체 단위로 버퍼 관리를 할 경우에는 GIS 클라이언트의 영역 질의에 대해서 GIS 서버는 클라이언트의 버퍼에서 관리되고 있는 객체를 제외한 결과를 보내주어야 하는데, GIS 서버는 GIS 클라이언트의 버퍼 내용을 알 수 없다. 따라서, 영역 질의에 대해서 객체 단위의 버퍼 관리는 불가능하다. 페이지 단위의 버퍼 관리도 마찬가지 문제가 발생한다. 즉, 개방형 GIS 컴포넌트에서 클라이언트와 서버간에는 표준 인터페이스만을 가지고 데이터 접근이 이루어지기 때문에, 서버와 클라이언트는 표준에 정의되지 않은 정보의 교환이 불가능하다.

따라서, 버퍼 관리의 단위는 클라이언트에서 요청한 영역 질의의 영역이 되어야 한다. (그림 3)은 클라이언트에서 요청한 영역 단위의 버퍼 관리 방법을 보이고 있다. (그림 3)(a)의 방법은 클라이언트에서 영역 질의를 수행할 때, 현재 버퍼에서 관리되고 있는 영역을 제외한 영역에 대해서 GIS 서버에 요청하고, 그 결과를 버퍼에서 관리하는 방법이다. 이 방법은 영역 질의에 대해서 현재 버퍼에 있는 영역을 제외한 영역을 임의 크기의 사각형 형태로 나누어야 하므로, 관리가 복잡해지는 문제점이 있다. 또한, 교체 전략 수립시 중첩된 영역에 대한 처리가 명확하지 않은 문제가 있다. 예를 들어, 그림에서 첫번째 영역과 두번째 영역에 중첩된 영역의 경우에는 LRU 상에서 두번째 영역의 데이터와 동일한 교체 우선순위를 가지고 있지만, 첫번째 영역의 데이터와 함께 관리되기 때문에, LRU 등의 교체전략 수립이 복잡해 지는 문제가 있다. (그림 3)(b)의 방법은 클라이언트에서 요청한 영역 질의에 해당하는 영역 전체를 버퍼에서 관리하는 방법이다. 이 방법은 버퍼에서 관리되는 데이터의 중복이 크며, 동일한 영역에 대한 질의가 요청되지 않을 경우에 버퍼의 활용도가 떨어지는 문제가 있다.



(a) 버퍼관리 방법 1 (b) 버퍼관리 방법 2

(그림 3) 클라이언터에서 요청한 영역 단위의 버퍼 관리 방법

GIS 클라이언트가 요청한 영역 단위의 버퍼 관리의 문제점을 해결하기 위해서 이 논문에서는 다음과 같은 방법을 사용한다. 첫째, 전체 데이터 영역을 일정한 크기로 분할하여, 각각의 분할된 영역을 버퍼 관리의 단위로 사용한다. 즉, 전체 데이터 영역에 대해서 고정 그리드 파일을 구성하고, 각각의 분할된 영역인 그리드 셀 단위로 버퍼를 관리한다. 고정 그리드 파일을 사용하는 이유는 개방형 GIS 컴포넌트에서 GIS 클라이언트는 GIS 서버의 데이터의 분포를 알 수 없으므로, 데이터 분포를 고려한 공간 분할 방법을 사용할 수 없기 때문이다. 둘째, GIS 클라이언트의 영역 질의에 대해서, 질의 영역과 겹치는 모든 그리드 셀 각각에 대한 영역 질의로 바꾸어 GIS 서버에 요청한다. 즉, 영역 질의의 대상인 클라이언트 화면 영역이 (그림 4)와 같다면, 이 영역과 겹치는 4개의 그리드 셀 각각에 대해서 4번의 영역 질의가 수행되고, 각 영역 질의 결과는 버퍼 관리의 대상이 된다.

(그림 4) 그리드 셀 단위의 버퍼 관리 방법

그리드 셀 단위의 버퍼 관리 방법은 (그림 3)에서 설명한 버퍼 관리의 문제점을 해결할 수 있다. 즉, 정형화된 크기의 그리드 셀을 버퍼 관리의 단위로 사용하기 때문에, 관리 및 교체전략 수립이 용이하며, 영역의 중복이 발생하지 않는 장점이 있다. 이 논문에서는 여러 그리드 셀에 중첩되는 공간 데이터는 각각의 그리드 셀에 중복해서 저장하는 방법을 사용한다.

3.2 영역 분할 단위

GIS 서버의 전체 데이터 영역을 일정한 크기로 분할하여 버퍼 관리를 할 경우에, 가장 중요한 점은 전체 영역을 분할하는 단위이다. 즉, GIS 클라이언트는 버퍼 관리의 단위가 되는 그리드 셀의 크기를 결정해야 한다. 효과적인 공간 분할을 위해서는 전체 데이터의 분포 및 크기에 대한 정보가 필요하지만, OGC의 표준 인터페이스를 준수하는 개방형 GIS 컴포넌트 소프트웨어에서 GIS 클라이언트는 GIS 서버의 (전체 데이터 영역, 전체 데이터 개수)만을 알 수 있을 뿐, 그 외의 메타 정보는 얻을 수 없다. 따라서, 이 논문에서는 균등 분포로 가정하고 영역을 분할하는 방법을 사용하며, 전체 데이터의 크기는 첫 영역 질의시 획득한 공간 데이터의 크기의 평균값으로 추정하는 방법을 사용한다. 즉, 첫번째 영역 질의 이후에 영역 분할이 이루어지게 된다.

전체 데이터 영역, 개수, 크기를 통해 영역을 분할하는

방법은 다음과 같다. 먼저, 이 논문에서는 MINSIZE를 설정해 두고, 하나의 그리드 셀내에 포함되는 공간 데이터의 크기가 최소한 MINSIZE보다 큰 경우에 영역을 분할한다. 이때, 영역의 분할은 고정 그리드 파일에서 영역 분할 방법과 동일하게 이루어 진다. 따라서, GIS 서버의 공간 데이터가 균등 분포일 경우에, 평균적으로 각 그리드 셀에 포함되는 공간 데이터의 크기는 MINSIZE 보다 작게 된다.

MINSIZE가 큰 경우에는 그리드 셀의 크기가 커지며, 반대로 MINSIZE가 작은 경우에는 그리드 셀의 크기가 작아지게 된다. 그리드 셀의 크기가 큰 경우에는 즉, 버퍼 관리의 단위가 큰 경우에는, GIS 클라이언트의 영역 질의와 중첩되는 그리드 셀의 개수가 적어진다. 따라서, GIS 서버로 요청되는 영역 질의의 회수가 줄어들게 되며, GIS 서버에서 공간 데이터의 검색 및 전송 비용이 줄어드는 장점이 있다. 또한 이 논문에서는 대용량 공간 데이터를 처리하기 위해서 메모리 뿐 아니라, 디스크 버퍼도 함께 사용하고 있기 때문에, 버퍼 관리의 단위가 큰 경우가 한번의 디스크 접근으로 많은 데이터에 접근할 수 있으므로 디스크 접근 비용도 줄어들게 된다. 왜냐하면, 전체 디스크 접근 비용은 탐색 비용(회전 지연 비용을 포함)과 전송 비용으로 나뉘는데, 탐색 비용이 전체 디스크 접근 비용의 많은 부분을 차지 한다. 기존의 연구[1, 20, 21]에서는 탐색 비용을 전송 비용의 50배, 10배, 그리고 15배로 각각 계산하고 있으며, [9]에 따르면 하드 디스크의 종류에 따라서 10배에서 40배의 차이가 나타난다. 특히, 고속의 하드 디스크일수록 전송 비용에 대한 탐색 비용의 비율이 높다. 따라서 버퍼 관리 단위를 크게 함으로서, 디스크 탐색 비용을 줄이는 것은 전체 질의 처리 비용을 줄이는데 크게 기여할 수 있다.

반면에, 버퍼 관리의 단위가 커질 경우에는 단편화(fragmentation) 문제가 발생하여, 전체 메모리 버퍼를 효과적으로 사용하지 못하는 문제가 발생한다. 단편화란 전체 메모리 버퍼에 일부 공간이 비어 있지만 하나의 그리드 셀이 들어가기에는 너무 작아서 사실상 사용될 수 없기 때문에 발생하는 메모리의 낭비를 의미한다. 따라서, 이 논문에서는 단편화 문제를 최소화하면서, 그리드 셀의 크기를 최대로 하도록 MINSIZE 설정한다. MINSIZE를 결정은 5장의 성능 평가 실험을 통해서 자세히 설명한다.

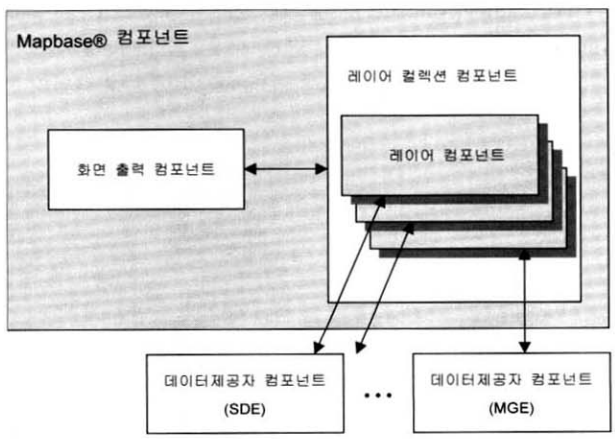
4. MapBase 컴포넌트에서 버퍼 관리 시스템의 설계

이 장에서는 개방형 GIS 컴포넌트 소프트웨어에서의 공간 데이터 접근 시간의 지연 문제를 해결하기 위한 버퍼 관리 시스템의 설계를 설명한다.

4.1 MapBase 컴포넌트

MapBase 컴포넌트는 한국전자통신연구원(ETRI, Electronics and Telecommunications Research Institute)에서 개발한 OGIS 데이터소비자로써, 공간데이터 접근 컴포넌트, 화면출력 컴포넌트, 공간데이터 모델링 컴포넌트, 공간 연산 컴포넌트, 좌표계 변환 컴포넌트, 공간 분석 컴포넌트, 공간데이터 편집 컴포넌트 등 다양한 기능을 수행하는 컴포넌트로 구성되어 있다. 현재, GeoMania, ZEUS, SDE, Shape 파일, SQL Server, MGE/GeoMedia, MapInfo, Small World 등의 GIS 서버에 대해서, MapBase 컴포넌트를 통해 접근될 수 있도록 OGIS 데이터제공자가 구현되어 있다.

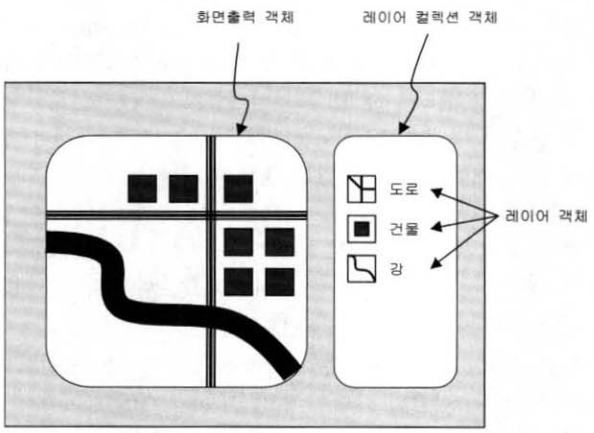
이 논문의 목적은 MapBase 컴포넌트의 성능 개선을 위하여 공간 데이터의 버퍼 관리 방버의 설계 및 구현이므로, 이 절에서는 MapBase 컴포넌트 중에서 공간 데이터에 접근하기 위해 사용되는 컴포넌트를 중심으로, MapBase 컴포넌트의 구성을 단순화하여 설명한다. (그림 5)는 MapBase 컴포넌트를 구성하는 컴포넌트들을 보이고 있다. 레이어 컬렉션 컴포넌트는 여러 개의 레이어 컴포넌트를 관리하는 컴포넌트이다. 각각의 레이어 컴포넌트는 OGIS 데이터제공자와의 연결을 통해서 GIS 서버에 있는 공간 데이터에 접근하는 역할을 담당하고 있으며, 레이어 컬렉션 컴포넌트 내에서 출력 우선순위(Display Priority)를 갖는다. 화면 출력 컴포넌트는 출력 우선순위에 따라서 선택된 레이어의 공간 데이터를 설정된 스타일로 화면에 출력하는 역할과 확대(Zoom-in), 축소(Zoom-out), 이동(Pan) 등 사용자와의 상호작용을 담당한다.



(그림 5) MapBase 컴포넌트의 구성

(그림 6)은 MapBase 컴포넌트를 이용하여 개방형 GIS 클라이언트 화면을 구성한 예를 보이고 있다. 화면 출력 객체는 레이어 컬렉션 객체에 포함된 도로, 건물, 강 레이어 객체에 대해서 각각 공간 데이터 접근을 요청하고, 그 결과를 미리 설정된 스타일에 따라서 화면에 출력한다. 각각의 레이어 객체는 OGIS 데이터제공자를 통해서 서로 다른 GIS 서버의 공간 데이터에 접근할 수 있다. 즉, 개방형 GIS 구조의 MapBase 컴포넌트에서는 서로 다른 GIS 서버간의 공

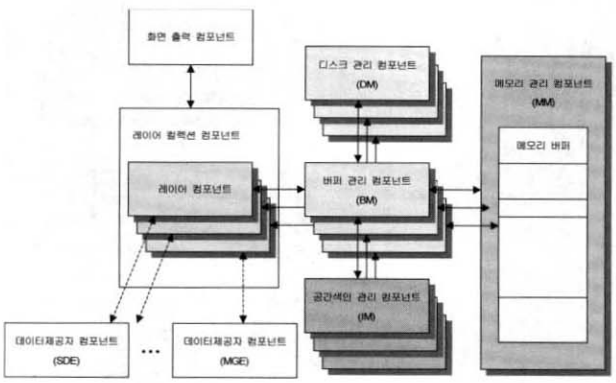
간 데이터를 상호 운용할 수 있으므로, OGIS 데이터제공자를 제공하는 모든 GIS 서버의 데이터를 공유할 수 있다. 예를 들어, 도로 레이어는 SDE에서, 건물 레이어는 ZEUS에서, 강 레이어는 MGE에서 접근하여 하나의 화면에서 출력, 조작, 연산, 분석 등이 가능하다.



(그림 6) MapBase 컴포넌트의 화면 구성 예

4.2 버퍼 관리를 위한 MapBase 컴포넌트의 확장

이 논문에서 화면 출력을 위해 접근된 공간 데이터를 메모리 및 디스크 상에 유지함으로써 사용자 응답 시간을 줄이기 위해서 MapBase 컴포넌트를 확장한 구조는 (그림 7)와 같다. 기존의 MapBase 컴포넌트에서 버퍼 관리 컴포넌트(BM, Buffer Manager), 디스크 관리 컴포넌트(DM, Disk Manager), 색인 관리 컴포넌트(IM, Index Manager), 메모리 관리 컴포넌트(MM, Memory Manager) 등이 추가되었다.



(그림 7) 메모리 관리를 위한 컴포넌트 구조

레이어 컬렉션 컴포넌트는 다수의 레이어 컴포넌트를 포함하고 있으며, 각 레이어 컴포넌트는 데이터제공자 컴포넌트를 통해 접근되는 하나의 레이어를 대표한다. 각 레이어는 자신의 연결 정보를 관리하기 위한 레이어 컴포넌트 이외에, BM 컴포넌트, DM 컴포넌트, IM 컴포넌트를 각각 하나씩 가진다. BM 컴포넌트는 데이터제공자 컴포넌트로부터 접근된 공간 데이터를 메모리 및 디스크 버퍼에 저장관리하기 위한 컴포넌트이다. 만약에 메모리 버퍼가 부족한 경우에 BM 컴포넌트는 DM 컴포넌트를 이용해서 디스크 버퍼에 저장하게 된다. IM 컴포넌트 해당 레이어에 대해서 고정 그리드 파일을 유지, 관리하는 역할을 담당한다.

사용자의 요청에 따라 화면 출력될 내용이 변경된 경우에, 화면 출력 컴포넌트는 출력 설정된 모든 레이어 컴포넌트를 통해 화면에 출력할 공간 데이터를 요청하게 된다. 이때, 레이어 컴포넌트는 자신과 연결 설정된 데이터제공자 컴포넌트 또는 BM 컴포넌트를 통해 공간 데이터에 접근한다. 화면 출력될 내용이 변경되는 경우는 크게 두 가지로 나뉜다. 첫째, 새로운 레이어가 추가되는 경우와, 둘째, 화면 영역의 변경(이동, 확대, 축소), 스타일 변경, 출력 우선순위의 변경, 출력 설정(출력 On/Off) 변경 등 기존 레이어의 출력 상태가 변경되는 경우이다. 새로운 레이어가 추가되는 경우, 즉, 해당 레이어에 처음 연결된 경우에 레이어 컴포넌트는 해당 데이터제공자 컴포넌트로부터 공간 데이터를 모두 읽어온 후에, BM 컴포넌트를 통해 메모리 및 디스크 버퍼에 저장관리를 요청한다. 반면에 기존 레이어의 출력 상태가 변경되는 경우에는 데이터제공자 컴포넌트가 아니라, BM 컴포넌트에 공간 데이터를 요청하게 된다. 이때, BM 컴포넌트는 데이터 요청을 처리하기 위해, 메모리 버퍼 또는 DM 컴포넌트가 관리하는 디스크 버퍼를 탐색한다. MM 컴포넌트는 레이어 수에 관계없이 전체적으로 하나만 생성되며, 레이어 컬렉션 컴포넌트 내에서 관리되는 모든 레이어에서 접근된 공간 데이터를 하나의 메모리 버퍼에서 관리하기 위해서 사용된다.

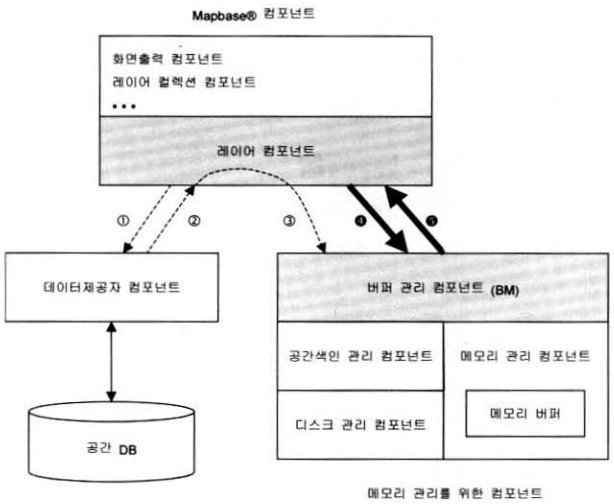
5. 구현 및 성능 평가

이 장에서는 개방형 GIS 컴포넌트 소프트웨어인 MapBase 컴포넌트에서 버퍼 관리 시스템의 구현과 메모리 및 디스크 버퍼 관리를 통한 데이터 접근 속도의 개선을 실험을 통해 평가한다.

5.1 구 현

(그림 8)은 MapBase 컴포넌트의 데이터 접근 구조를 보이고 있으며, MapBase 컴포넌트의 레이어 컴포넌트가 데이터제공자 컴포넌트 또는 BM 컴포넌트에 데이터를 요청하는 것을 의미한다. 버퍼 관리를 위해 추가 구현된 컴포넌트는 기존 MapBase 컴포넌트와의 인터페이스가 최소화되도록 설계되었다. 즉, 레이어 컴포넌트는 메모리 및 디스크 버퍼에서 관리되는 공간 데이터에 접근하기 위해서 BM 컴포넌트의 인터페이스만을 이용하며, IM 컴포넌트, DM 컴포넌트, MM 컴포넌트의 인터페이스를 직접 사용하지는 않는다. 그리고 MINSIZE 및 MAXSIZE 상수 값을 변경할 수

있도록 구현함으로써, 응용 및 공간 데이터의 특성에 따라 버퍼의 크기를 조절할 수 있도록 하였다. MINSIZE는 3.2절에서 설명하였으며, MAXSIZE는 메모리 버퍼가 관리하는 최대의 메모리 크기를 의미한다.



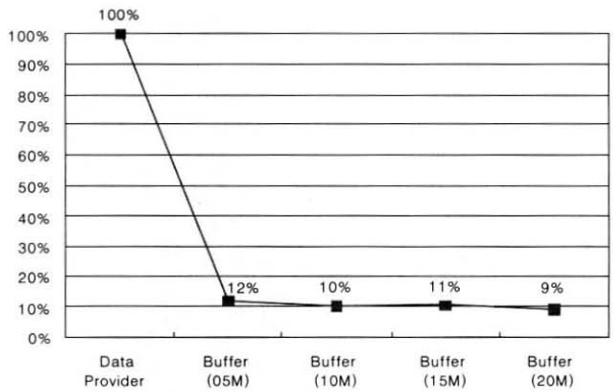
(그림 8) MapBase 컴포넌트의 데이터 접근 구조

기존 컴포넌트와 추가된 컴포넌트간의 인터페이스를 최소화하여 구현함으로써 얻는 장점은 다음과 같다. 첫째, 새로 추가된 컴포넌트를 기존 MapBase 컴포넌트에 통합하는 것이 용이하다. 둘째, IM 컴포넌트, DM 컴포넌트, MM 컴포넌트의 변경이 용이하다. 예를 들어, 버퍼 관리를 위해 수정된 레이어 컴포넌트의 코드를 변경하지 않고, MM 컴포넌트에서 메모리 버퍼의 교체 전략을 변경할 수 있다.

5.2 성능 평가

이 절에서는 OGIS 데이터소비자에서 메모리 관리를 통한 데이터 접근 속도의 개선을 실험을 통해 입증한다. 실험은 512Mbyte 메모리의 펜티엄III-1GHz 환경에서 수행하였으며, 메모리 관리를 위한 컴포넌트는 C++로 구현되었으며, Windows 2000에서 Visual C++ 6.0로 컴파일 하였다.

첫번째 실험에서는 데이터제공자 컴포넌트에서 데이터 모델 변환으로 인해 매우 많은 시간이 걸리므로, 메모리 버퍼에 비해 상대적으로 느린 디스크 버퍼를 통해서도 데이터 접근 시간이 개선될 수 있음을 보인다. (그림 9)는 데이터제공자 컴포넌트를 통해 공간 데이터에 접근하는 시간과 디스크 버퍼에 있는 데이터에 접근하는 시간을 비교하여 보여주고 있다. 실험은 25Mbyte 크기의 공간 데이터를 갖는 레이어를 사용하였다. 그림의 가로축에서 Data Provider는 데이터제공자 컴포넌트를 통해 공간 데이터를 읽어오는 것을 의미하며, Buffer(SIZE)는 DM 컴포넌트를 통해 디스크에 저장되어 있는 공간 데이터를 읽어오는 것을 의미한다. SIZE는 MINSIZE의 크기로써, 각각 MINSIZE가 5, 10, 15, 20Mbyte임을 나타낸다. MINSIZE가 크다는 것은 그리드 셀의 크기가 크다는 것이며, 디스크 버퍼에서 한번에 읽어오는 공간 데이터의 양이 더 많다는 것을 의미한다. 그림의 세로축은 데이터제공자 컴포넌트를 통해 공간 데이터에 접근하는 시간에 대해 DM 컴포넌트의 디스크 버퍼를 통해 접근하는 시간의 비율을 의미한다. 실험 결과 디스크 버퍼에 저장된 공간 데이터를 읽어오는 시간에 비해서 데이터제공자 컴포넌트를 통해서 공간 데이터를 읽어오는 시간이 훨씬 오래 걸린다는 것을 알 수 있다. 즉, 데이터제공자 컴포넌트 내부에서 ① GIS 서버에 접속하고, ② 해당 GIS 서버의 고유의 공간 데이터 모델에 의해 표현된 공간 데이터를 읽어오고, ③ 이 데이터를 WKB 형태의 공간 데이터로 변환하는 시간이 WKB 형태로 디스크 버퍼에 저장된 공간 데이터에 접근하는 시간보다 매우 길다는 것을 알 수 있다. (그림 9)의 결과는 메모리 버퍼를 사용하지 않고, 디스크 버퍼를 사용하는 경우에도 데이터제공자 컴포넌트만을 사용할 경우보다 90% 정도의 성능 개선을 보이고 있다. 또한 MINSIZE를 크게 하면, 디스크에서의 데이터 접근 비용이 적어짐을 보이고 있다.

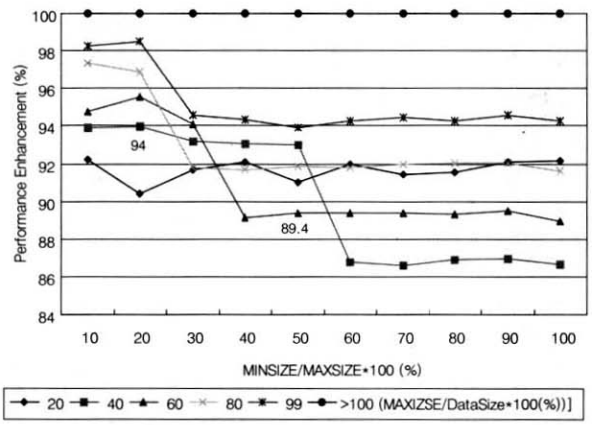


(그림 9) 데이터 접근 시간 비교

두번째 실험은 디스크 버퍼뿐 아니라, 메모리 버퍼를 사용하는 경우의 성능 평가를 비교한다. 특히, MAXSIZE, MINSIZE와 성능 개선과의 관계를 비교한다. MAXSIZE는 MM 컴포넌트에서 관리하는 전체 메모리 버퍼의 크기하며, MINSIZE는 그리드 셀의 크기를 결정하는데 사용된다. (그림 10)은 MAXSIZE, MINSIZE에 따라 메모리 버퍼의 성능을 비교하여 보이고 있다. 이 실험에서는 5Mbyte 크기의 공간 데이터를 갖는 레이어 5개로 구성되어, 총 25Mbyte(=Data Size)의 크기의 갖는 데이터 셋을 사용하였다. (그림 11)에서는 동일한 데이터 셋에 대해서, MAXSIZE를 전체 데이터 셋 크기(DataSize)의 20%, 40%, 60%, 80%, 99%, 100% 이상으로 구분하여 실험한 결과를 보이고 있다. 그림의 가로축은 MAXSIZE 크기에 대한 MINSIZE 크기의 비율을 의미한다. 즉, 가로축의 10는 MINSIZE가 주어진 MAXSIZE

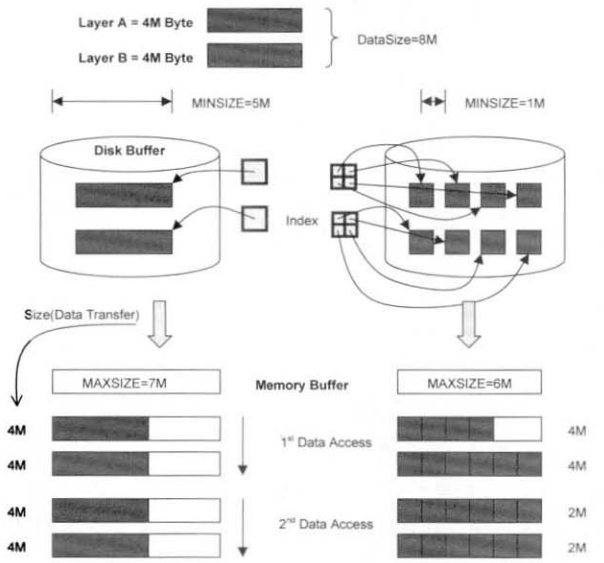
의 10% 크기임을 의미한다.

MAXSIZE가 100% 이상이라는 것은 데이터 셋의 모든 공간 데이터를 메모리 버퍼에서 관리할 수 있음을 의미한다. 따라서, 공간 데이터를 메모리에 적재하기 위한 데이터 접근 시간이 필요하지 않기 때문에, 데이터제공자 컴포넌트를 통해 공간 데이터에 접근하는 시간에 대한 성능 개선은 100%이다. MAXSIZE가 크다는 것은 MM 컴포넌트가 관리하는 전체 메모리 버퍼의 크기가 크다는 것을 의미하므로, MAXSIZE가 크면 클수록 성능이 더욱 좋아질 것으로 예상된다. 그러나, MAXSIZE가 더 작더라도 MINSIZE의 크기에 따라서 성능이 좋은 경우가 발생한다. 예를 들어, MAXSIZE가 DataSize의 60%일때, MINSIZE를 MAXSIZE의 50%로 설정한 경우의 성능 개선 비율은 89.4%이지만, MAXSIZE가 DataSize의 40%이더라도, MINSIZE를 MAXSIZE의 20%로 설정한 경우의 성능 개선 비율은 94%가 된다.



(그림 10) MAXSIZE, MINSIZE에 따른 성능 비교

MAXSIZE가 더 작더라도 MINSIZE의 크기에 따라서 성능이 좋은 경우가 발생하는 이유는 MINSIZE가 그리드 셀의 크기에 영향을 주기 때문이다. 그리드 셀은 디스크 버퍼와 메모리 버퍼간의 데이터 전송의 단위로 사용되기 때문에, 그리드 셀이 큰 경우에는 디스크 접근 횟수가 적어지는 대신에 전송되는 데이터의 양이 커지며, 메모리 버퍼에서 단편화가 발생할 확률이 커지게 된다. (그림 11)은 MINSIZE에 의해 결정된 그리드 셀의 크기와 성능과의 관계를 보이고 있다. 4M Byte의 두 레이어 A, B에 대해서, (그림 11)(a)는 7M의 메모리 버퍼, (그림 11)(b)는 6M의 메모리 버퍼를 가진다. 버퍼 관리를 위해 데이터제공자 컴포넌트로부터 공간 데이터에 접근할 때, (그림 11)(a)의 경우에는 5M의 MINSIZE를 가지므로 하나의 그리드 셀을 갖는 고정 그리드 파일이 생성된다. 반면에 1M의 MINSIZE를 갖는 (그림 11)(b)의 경우에는 최소한 4개 이상의 그리드 셀을 갖는 고정 그리드 파일이 생성된다. 이와 같이 생성된 고정 그리드 파일 및 디스크 버퍼, 메모리 버퍼를 사용하여 공간 데이터를 계속해서 접근할 경우에 발생되는 비용은 메모리 버퍼의 단편화로 인해서 MAXSIZE가 더 큰 (그림 11)(a)에서 더욱 커짐을 알 수 있다. (그림 11)(a)에서는 두 번의 공간 데이터 접근을 위해서 모두 16M의 데이터 전송이 발생되며, (그림 11)(b)에서는 12M의 데이터 전송이 발생된다. 즉, 상대적으로 많은 메모리 버퍼가 할당되어 있어도, 사용되지 않는 메모리가 많게 되면 좋은 성능을 보일 수 없다.



(a) MAXSIZE = 7M, MINSIZE = 5M

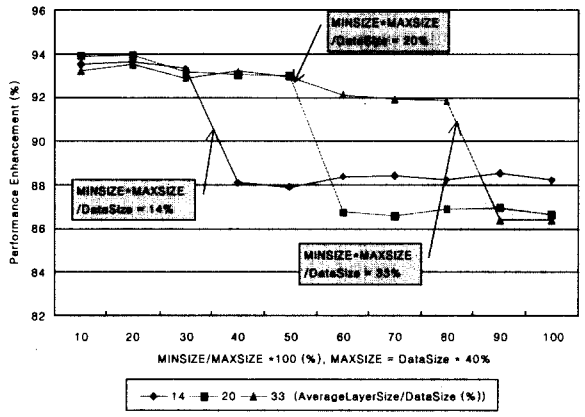
(b) MAXSIZE = 6M, MINSIZE = 1M

(그림 11) MINSIZE와 그리스 셀의 크기에 따른 데이터 전송량 및 단편화

(그림 11)의 실험 결과를 정리하면 MINSIZE, MAXSIZE를 결정하는 기준을 알 수 있다. 첫째, MAXSIZE는 전체 메모리 버퍼의 크기로써, 가능한 범위에서는 가장 크게 하는 것이 좋다. 만약, 접근해야 할 전체 데이터셋의 크기보다 크다면 디스크 버퍼를 사용하지 않아도 된다. 둘째, MINSIZE는 전체 데이터 셋에 포함된 레이어 들의 평균 크기보다는 작게 하는 것이 좋다. 즉, MINSIZE를 줄임으로써, 메모리 버퍼의 단편화를 줄임으로서, MAXSIZE 만큼의 전체 메모리를 모두 활용될 수 있도록 한다.

(그림 12)는 MINSIZE의 크기와 성능과의 관계에 대한 (그림 11)의 실험 결과가 서로 다른 데이터 셋에 대해서도 성립함을 보이고 있다. 이 실험에서는 각 레이어의 평균 크기가 전체 데이터 셋 크기의 14%, 20%, 33%인 세 가지 데이터 셋을 사용하였다. MAXSIZE는 모두 전체 데이터 셋 크기의 40%로 고정하였으며, 가로축을 의미하는 MINSIZE는 MAXSIZE의 10%~100%로 변경시켜 실험하였다. 각각의 데이터 셋은 유사한 형태의 실험 결과를 보이고 있다. 즉, 특정한 크기의 MINSIZE 전후로 성능 향상 비율이 급

격한 차이를 보이고 있으며, 급격한 성능 차이를 보인 MINSIZE 전후를 제외하면 MINSIZE의 변화에 대해서 성능의 차이는 거의 보이지 않는다. 각각의 데이터 셋에 대해서 급격한 성능 차이를 보인 MINSIZE는 각 데이터 셋에 속한 각 레이어의 평균 크기와 거의 일치함을 알 수 있다. 즉, MINSIZE는 전체 데이터 셋에 포함된 레이어 들의 평균 크기보다는 작게 하는 것이 좋다는 (그림 11)의 결과가 서로 다른 데이터 셋에 대해서도 적용됨을 보이고 있다.



(그림 12) MINSIZE, MAXSIZE, DataSize와의 관계 비교

6. 결론 및 향후 연구 과제

개방형 GIS 구조에서는 서로 다른 GIS 서버에서, 서로 다른 데이터 모델에 따라 구축된 공간 데이터를 동일한 방법으로 접근하기 위한 표준 인터페이스를 제공한다. 표준 인터페이스는 상호 운용성을 지원하지만, 반면에 데이터 모델의 변환에 따른 데이터 접근 속도의 저하를 초래한다.

이 논문에서는 공간 데이터 접근 속도를 개선하기 위해 OGIS 데이터소비자를 위한 메모리 관리 방법을 설계하고 구현하였다. OGIS 데이터제공자는 ETRI에서 개발한 Map-Base 컴포넌트를 사용하고 있으며, 메모리 관리를 위해서 버퍼관리 컴포넌트, 디스크관리 컴포넌트, 공간색인관리 컴포넌트, 메모리관리 컴포넌트를 구현하였다. 제안된 메모리 관리 기법은 하나 이상의 레이어를 고려하였기 때문에, 각 레이어에 대해서 가변 길이의 메모리 버퍼를 지원함으로써 전체 메모리 버퍼를 효과적으로 활용할 수 있는 특징이 있다. 또한 동적으로 공간 색인을 생성관리함으로써 버퍼관리 컴포넌트를 통한 데이터 접근 속도를 개선하였다.

이 논문에서 구현된 메모리 관리 컴포넌트를 통해서 Map Base 컴포넌트의 데이터 접근 속도는 데이터제공자 컴포넌트만을 사용할 때보다 크게 개선되었음을 다양한 성능 평가 실험을 통해 입증하였다. 또한 실험 결과를 기반으로 메모리 관리 컴포넌트의 성능 개선을 위해서 MINSIZE, MAX SIZE의 결정을 위한 기준을 마련하였다. 향후, 레이어의 크기, 데이터 셋의 크기, 각 영역의 공간 데이터 접근 빈도에 대한 정보, 메모리 버퍼의 크기, 디스크 버퍼의 크기 등을 고려하여 최적화된 메모리 관리 기법에 대한 연구가 필요하다.

참 고 문 헌

[1] A. Hutflesz, H.-W. Siz and P. Widmayer, "Globally Order Preserving Multidimensional Linear Hashing," ICDE, p.572, 1988.

[2] A. Kemper and D. Kossmann, "Dual-Buffering Strategies in Object Bases," In Proceedings of the 20th Conference on Very Large Data Bases, September, 1994.

[3] E. G. Coffman, Jr. and P. J. Denning, "Operating Systems Theory," Prentice-Hall, 1973.

[4] E. J. ONeil, P. E. ONeil and G. Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering," In Proceedings of the ACM SIGMOD International Conference on Management of Data, May, 1993.

[5] G. Droege and H. J. Scheck, "Query-adaptive data space partitioning using variable-size storage clusters," Advances in Spatial Databases, Springer-Verlag, pp.337-356, 1993.

[6] J.-H. Ahn and H.-J. Kim, "SEOF : An Adaptable Object Prefetch Policy For Object-Oriented Database Systems," Thirteenth International Conference on Data Engineering, April, 1997.

[7] J. Nievergelt, H. Hinterberger, K. C. Sevcik, "The Grid File : An Adaptable, Symmetric Multikey File Structure," ACM Trans. On Database Systems, Vol.9, No.1, pp.38-71, 1984.

[8] K. Wilkinson, M. Neimat, "Maintaining Consistency of Client-Cached Data," In Proceedings of the 16th Conference on Very Large Data Bases, Brisbane, 1990.

[9] M. J. Folk, B. Zoellick and G. Riccardi, "File Structures : An Object-Oriented Approach with C++," Addison Wesley, p.49, 1998.

[10] M. Palmer and S. B. Zdonik, "Fido : A Cache That Learns to Fetch," In Proceedings of the 17th International Conference on Very Large Data Bases, September, 1991.

[11] T. Johnson and D. Shasha, "2Q : A Low Overhead High Performance Buffer Management Replacement Algorithm," In Proceedings of the 20th Conference on Very Large Data Bases, September, 1994.

[12] OpenGIS Consortium Inc., The OpenGIS Simple Feature Specification for OLE/COM Revision 1.1, 1999.

[13] E. E. Chang and R. H. Katz, "Exploiting Inheritance and Structure Semantics for Effective Clustering and Buffering in an Object-Oriented DBMS," In Proceedings of the ACM SIGMOD International Conference on Management of Data, May, 1989.

[14] R. Alonso, D. Barbara, H. Garcia Molina, "Data Caching Issues in an Information Retrieval System," ACM Transactions on Database Systems, Vol.15, No.3, September, 1990.

[15] K. Wilkinson, M. Neimat, "Maintaining Consistency of Client-Cached Data," In Proceedings of the 16th Conference on Very Large Data Bases, Brisbane, 1990.

[16] James E. Pitkow and Margaret M. Recker, "A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns," Electronic Proceedings of the Second World Wide Web Conference '94 : Mosaic and the Web, 1994.

[17] Michael J. Franklin, Michael J. Carey and Miron Livny, "Local Disk Caching for Client-Server Database Systems," In Proceedings of the 19th Conference on Very Large Data Bases, Dublin, 1993.

[18] Michael J. Franklin and Michael J. Carey, "Client-Server Caching Revisited," In Proceedings of the International Workshop on distributed Object Managemnt, 1992.

[19] S. Dar, M. Franklin, B. T. Jonsson, D. Srivastava and M. Tan, "Semantic Data Caching and Replacement," In Proceedings of VLDB Conference, 1996.

[20] T. Brinkhoff and H. P. Kriegel, "The Impact of Global Clustering on Spatial Database Systems," VLDB, p.168, 1994.

[21] T. Brinkhoff, H. Horn, H. P. Kriegel and R. Schneider, "A Storage and Access Architecture for Efficient Query Processing in Spatial Databases," Proc. 3rd Int. Symp. On Large Spatial Database, Singapore, pp.357-376, 1993.

조 대 수

e-mail : junest@etri.re.kr

1995년 부산대학교 컴퓨터공학과(학사)

1997년 부산대학교 컴퓨터공학과(공학석사)

2001년 부산대학교 컴퓨터공학과(공학박사)

2001년~현재 한국전자통신연구원 LBS 연구팀 선임연구원

관심분야 : GIS, 공간 DB, LBS, 이동체 DB 등

민 경 욱

e-mail : jhp@etri.re.kr

1996년 부산대학교 전자계산학과(학사)

1998년 부산대학교 전자계산학과(석사)

2001년~현재 한국전자통신연구원 LBS 연구팀 연구원

관심분야 : 공간데이터베이스, 이동체데이터베이스, LBS, GIS 등